Spring 4-1-1985

Boundary NLC Graph Grammars Basic Definitions, Normal Forms and Complexity ; CU-CS-293-85

Grzegorz Rozenberg
University of Leiden

Welzl
University of Colorado Boulder

BOUNDARY NLC GRAPH GRAMMARS

BASIC DEFINITIONS, NORMAL FORMS, AND COMPLEXITY

by

G. Rozenberg and E. Welzl

CU-CS-293-85 April, 1985

*Insitute of Applied Mathematics and Computer Science, University of Leiden, Leiden, The Netherlands.

Report Nr. 84-29

BOUNDARY NLC GRAPH GRAMMARS
BASIC DEFINITIONS, NORMAL FORMS, AND COMPLEXITY

by

Grzegorz Rozenberg and Emo Welzl*)

November, 1984

Institute of Applied Mathematics and Computer Science

University of Leiden

Wassenaarseweg 80, Leiden

The Netherlands

*) On leave from: Institutes for Information Processing IIG, Technical University of Graz and Austrian Computer Society, Schiesstattgasse 4a, A-8010 Graz, AUSTRIA.

Node label controlled (NLC) grammars are graph grammars (operating on node labeled undirected graphs) which rewrite single nodes only and establish connections between the embedded graph and the neighbors of the rewritten node on the basis of the labels of the involved nodes only. They define (possibly infinite) languages of undirected node labeled graphs (or, if we just omit the labels, languages of unlabeled graphs).

Here we consider a restriction of NLC grammars, so-called boundary NLC (BNLC) grammars, distinguished by the property that whenever in a graph already generated two nodes may be rewritten, then these nodes are not adjacent. The graph languages generated by this type of grammars are called BNLC languages.

Although we show that this restriction leads to a smaller class of languages, still enough generative power remains to define interesting graph languages. For example, trees, complete bipartite graphs, maximal outerplanar graphs, k-trees, graphs of bandwidth $\leq k$, graphs of cyclic bandwidth $\leq k$, graphs of binary tree bandwidth $\leq k$, graphs of cutwidth $\leq k$, (always for a fixed positive integer k) turn out all to be BNLC languages.

We prove a number of normal forms for BNLC grammars and then we indicate their usefulness by various applications. In particular, we show that for connected graphs of bounded degree the membership problem for BNLC languages is solvable in deterministic polynomial time.

INTRODUCTION

Node label controlled (NLC) grammars are graph grammars operating on node labeled undirected graphs. A production in an NLC grammar is a pair (d,Y), where d is a label and Y is a graph. Such a production is applicable to a node x in a graph X if and only if x is labeled by d. The rewriting process consists of (i) deleting x in X (together with incident edges), (ii) adding Y disjointly to the remainder of X and (iii) establishing connections between nodes in Y and ("former") neighbors of x in the remainder of X. This embedding is controlled by a so-called connection function conn which maps labels to sets of labels. More specifically, a neighbor z (of x) labeled by a is connected to a node y (of Y) labeled by b if and only if $a \in conn(b)$. The graph language generated by an NLC grammar consists of the set of all graphs such that (i) they can be obtained from the axiom (graph) Z_{ax} of the grammar by a sequence of rewritings, and (ii) they have labels only from the set Δ of terminal labels of the grammar.

NLC grammars have been introduced by Janssens & Rozenberg (1980a,b) as a basic framework for the mathematical investigation of graph grammars (the more general work on the theory of graph grammars is well presented in Nagl, 1979, and Ehrig, 1979). Since then this model has been intensively investigated, see e.g. Janssens & Rozenberg (1981), Brandenburg (1983), Turán (1983), Ehrenfeucht et al. (1984) and Janssens et al. (1984). In particular, it has turned out that most basic problems of graph theoretic nature concerning NLC grammars (languages) are undecidable. Although the membership problem for NLC grammars is decidable, NLC grammars can generate PSPACE-complete graph languages. Results like this have inspired a search for feasible but "nontrivial" subclasses of the class of NLC grammars (see e.g. Janssens, 1983).

The class of boundary NLC grammars, BNLC grammars for short,is defined as follows. An NLC grammar is a BNLC grammar if (i) the left-hand side of each production is a nonterminal label, and (ii) all the graphs involved (i.e., the axiom and the right-hand sides of productions) are such that two nonterminally labeled nodes are never adjacent. It turns out that the class of BNLC languages (i.e., the graph languages generated by BNLC grammars) can be defined by the subclass of NLC grammars in which (i) the left-hand side of each production is a nonterminal label and (ii) the range of the connection function consists of terminal labels only. Hence, on the one hand one can view BNLC grammars as an analogue (in the framework of NLC grammars) of fundamental subfamilies of context-free string grammars (such as linear grammars or context-free grammars in operator normal form), while, on the other hand, one gets a characterization of BNLC languages by considering a restric-

tion on NLC grammars that is certainly a very natural one from the mathematical point of view.

The family of BNLC grammars appears to be quite attractive from both the feasibility and the generative power point of view. Thus, e.g. (1) one can prove that membership of connected graphs of bounded degree in BNLC languages is decidable in polynomial time, while (2) quite a number of interesting families of graphs can be generated by BNLC grammars (e.g., trees, complete bipartite graphs, maximal outerplanar graphs, k-trees, graphs of bandwidth $\leq k$, graphs of cyclic bandwidth $\leq k$, graphs of binary tree bandwidth $\leq k$, graphs of cutwidth $\leq k$, always for a fixed positive integer k).

We would like to point out that this paper presents only some basic results concerning BNLC grammars and languages. We will report more results that justify the above contention in the future (see also the discussion in Section 7).

The paper is organized as follows. After recalling in Section 1 some preliminaries from graph theory and the theory of graph grammars, basic definitions and examples concerning BNLC grammars are given in Section 2. In Sections 3 through 5 we prove three normal form results and discuss some immediate implications of them. The main application of these normal forms are the complexity results presented in Section 6 where it is proved that for connected graphs of bounded degree the membership problem in BNLC languages is solvable in polynomial time. This holds even if the given graph is "unlabeled". This result appears to be quite on the edge between P and NP-complete, because already for (possibly disconnected) unlabeled graphs of bounded degree, as well as for connected unlabeled graphs of bounded average degree the corresponding membership problems are NP-complete.
Finally, Section 7 discusses the results obtained and gives some future perspectives.

1. PRELIMINARIES

We start with basic notations concerning graphs and graph grammars, which we need for this paper. We assume familiarity with rudimentary graph theory. In particular, we use the following notions as defined in Harary (1969): adjacent, neighbor, degree of a node in a graph, subgraph, induced subgraph, path in a graph, connected graph, complete bipartite graph.

Graphs

We consider finite undirected node labeled graphs without loops and without multiple edges. For a set of labels Σ, a graph X (over Σ) is specified by a finite set V_X of nodes, a set E_X of two element subsets of V_X (called the set of edges) and a function φ_X from V_X into Σ (called the labeling function). Disregarding the labeling function one gets the underlying unlabeled graph of X, denoted by $\underline{und}(X)$. The set of all graphs over Σ is denoted by G_Σ. The unique graph on the empty set of nodes is called empty graph and it is denoted by λ.

The label set of a graph X, denoted $\underline{lab}(X)$, is the set $\{\varphi_X(x) \mid x \in V_X\}$. For a finite set V, $\#V$ denotes its cardinality; for a graph X we often write $\#X$ rather than $\#V_X$.

For a graph X and a subset V of V_X, the neighborhood of V (in X), denoted $\underline{neigh}_X(V)$, is the set $\{y \in V_X - V \mid$ there is an $x \in V$ with $\{x,y\} \in E_X\}$. If $V = \{x\}$ then we write also $\underline{neigh}_X(x)$ instead of $\underline{neigh}_X(\{x\})$. The context of a node x (in a graph X), denoted $\underline{cont}_X(x)$, is the set $\{\varphi_X(y) \mid y \in \underline{neigh}(x)\}$.

For a graph X and a node $x \in V_X$, the graph $X-x$ is the subgraph of X induced by $V_X - \{x\}$.

Two graphs X and X' are isomorphic, if there is a bijection from V_X to $V_{X'}$ which preserves labels and adjacencies. The set of all graphs isomorphic to a given graph X is denoted by $[X]$.

Let Σ and Σ' be sets of labels. A relabeling ρ from Σ' to Σ is a function from

Σ' into Σ. Let X be a graph over Σ'. Then the ρ-image of X, denoted $\rho(X)$, is the graph defined by $V_{\rho(X)} = V_X$, $E_{\rho(X)} = E_X$ and $\varphi_{\rho(X)}(x) = \rho(\varphi_X(x))$ for all $x \in V_{\rho(X)}$.

For a graph language L we denote by und(L) the underlying unlabeled language of L, i.e., the set $\{\text{und}(X) \mid X \in L\}$.

Graph Grammars

A node label controlled (NLC) grammar is a system $G = (\Sigma, \Delta, P, \text{conn}, Z_{ax})$, where Σ is a finite nonempty set of labels, Δ is a nonempty subset of Σ (the set of terminals), P is a finite set of pairs (d,Y), with $d \in \Sigma$ and $Y \in G_\Sigma$ (the set of productions), conn is a function from Σ into 2^Σ (connection function) and $Z_{ax} \in G_\Sigma$ (axiom).

By $[P]$ we denote the abstract production set $\{(d,Y') \mid Y' \in [Y]$ for some $(d,Y) \in P\}$. The set $\Sigma-\Delta$ is referred to as the set of nonterminals and we will reserve the symbol Γ (possibly with an appropriate inscription) to denote $\Sigma-\Delta$.

In the context of G, given a graph $X \in G_\Sigma$ we refer to nodes labeled by elements of Γ (Δ, respectively) as nonterminal nodes (terminal nodes, respectively).

By maxr(G) we denote $\max(\{\#Z_{ax}\} \cup \{\#Y \mid (d,Y) \in P$ for some $d \in \Sigma\})$.

Let X,Y,Z be graphs in G_Σ with $V_X \cap V_Y = \emptyset$ and let $x \in V_X$. Then X concretely derives Z (in G, replacing x by Y), denoted by $X \underset{G}{\Rrightarrow}_{(x,Y)} Z$, if $(\varphi_X(x),Y) \in [P]$, $V_Z = V_{X-x} \cup V_Y$,

$$E_Z = E_{X-x} \cup E_Y \cup \{\{x',y\} \mid x' \in \text{neigh}_X(x),\ y \in V_Y,\ \varphi_X(x') \in \text{conn}(\varphi_Y(y))\},$$

φ_Z equals φ_{X-x} on V_{X-x}, and φ_Z equals φ_Y on V_Y. (If G is understood, then we write simply $X \Rrightarrow_{(x,Y)} Z$.) Intuitively, we replace x in X by the graph Y and connect a node y of Y to a neighbor x' of x if and only if $\varphi_X(x') \in \text{conn}(\varphi_Y(y))$. We will refer to X as the host graph, x as the mother node, $X-x$ as the remainder (graph), Y as the daughter graph, and Z as the resulting graph.

A graph X directly derives a graph Z (in G), in symbols $X \underset{G}{\Rightarrow} Z$, if there is a graph $Z' \in [Z]$, such that X concretely derives Z'. $\underset{G}{\overset{\star}{\Rightarrow}}$ is the transitive and re-

flexive closure of $\underset{G}{\Rightarrow}$. If $X \overset{*}{\underset{G}{\Rightarrow}} Z$, then we say that X derives Z (in G).

The exhaustive language, S(G), of G is the set $\{X \in G_\Sigma \mid Z_{ax} \overset{*}{\underset{G}{\Rightarrow}} X\}$ and the language, L(G), of G is the set $\{X \in G_\Delta \mid Z_{ax} \overset{*}{\underset{G}{\Rightarrow}} X\}$.

A graph language L is an NLC language if there is an NLC grammar G such that L = L(G). A language L of unlabeled graphs is an unlabeled NLC (u-NLC) language, if there is an NLC language L' such that L = und(L').

A nonterminal NLC (NNLC) grammar is an NLC grammar $G = (\Sigma,\Delta,P,\underline{conn},Z_{ax})$, where $(d,Y) \in P$ implies that $d \in \Gamma$; it is known that for every NLC grammar G there is an NNLC grammar G' with L(G') = L(G), see Janssens and Rozenberg (1980b).

Remark 1.1. Our definition of an NLC grammar differs from the original one of Janssens & Rozenberg (1980a) in two points. First, for technical reasons we use a connection function conn: $\Sigma \to 2^\Sigma$ instead of a connection relation $C \subseteq \Sigma \times \Sigma$. (The correspondence is given by the standard function-relation correspondence.) Secondly, our definition is slightly more general than the original, in that it allows the right-hand side to be the empty graph λ. However, as pointed out also in Ehrenfeucht et al. (1984), this difference is not essential. □

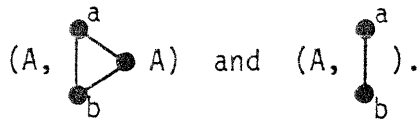
2. DEFINITIONS AND BASIC PROPERTIES

Let Φ be a set of labels. A graph X is called a Φ-boundary graph, if no two nodes of X that are labeled by elements of Φ are adjacent.

A boundary NLC (BNLC) grammar is an NNLC grammar $G = (\Sigma,\Delta,P,\underline{conn},Z)$, where Z is a Γ-boundary graph and, for all $(d,Y) \in P$, Y is a Γ-boundary graph. A graph language L is a BNLC language, if there is a BNLC grammar G such that $L = L(G)$. A language L of unlabeled graphs is an unlabeled BNLC (u-BNLC) language, if there is a BNLC language L' such that $L = \underline{und}(L')$. (Recall, that we set implicitly $\Gamma = \Sigma - \Delta$!).

We start our considerations by giving some examples of BNLC grammars and BNLC languages.

Example 1 (complete bipartite graphs). Consider the BNLC grammar $G_1 = (\Sigma_1,\Delta_1,P_1,\underline{conn}_1,Z_1)$, where $\Delta_1 = \{a,b\}$, $\Sigma_1 = \{A\} \cup \Delta_1$, Z_1 is a single node labeled by A, $\underline{conn}_1(A) = \Delta_1$, $\underline{conn}_1(a) = \{b\}$, $\underline{conn}_1(b) = \{a\}$, and P_1 consists of the following two productions:



Then $\underline{und}(L(G_1))$ consists of all complete bipartite graphs of the form $K_{n,n}$, $n \geq 1$. Fig. 2.1 shows a derivation step in G_1.

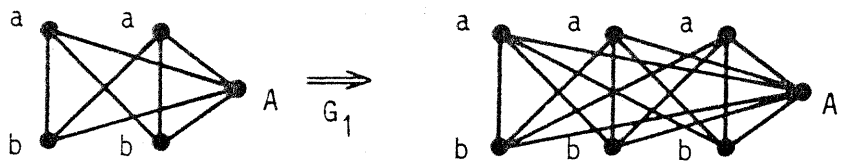


Fig. 2.1. A derivation step in G_1.

It is easily seen that by adding productions $(A, \overset{a}{\bullet}\!\!-\!\!\bullet^{A})$ and $(A, \overset{b}{\bullet}\!\!-\!\!\bullet^{A})$ to G_1 one obtains a BNLC grammar G_1' such that $\underline{und}(L(G_1'))$ equals the set of all complete bipartite unlabeled graphs $K_{n,m}$, $n,m \geq 1$. □

Example 2 (maximal outerplanar graphs). Let $G_2 = (\Sigma_2, \Delta_2, P_2, \underline{conn}_2, Z_2)$ with $\Delta_2 = \{a,b,c\}$, $\Sigma_2 = \{A,B,C\} \cup \Delta_2$, $\underline{conn}_2(A) = \{b,c\}$, $\underline{conn}_2(B) = \{a,c\}$, $\underline{conn}_2(C) = \{a,b\}$, and $\underline{conn}_2(a) = \underline{conn}_2(b) = \underline{conn}_2(c) = \Delta_2$. P_2 and Z_2 are depicted in Figs. 2.2 and 2.3, respectively. Following a simple characterization of maximal outerplanar graphs in Proskurowski (1979), it can be seen that $L(G_2)$ consists of all maximal outerplanar graphs labeled by Δ_2, where no two adjacent nodes have the same label. (A maximal outerplanar graph is a triangulation of a polygon, see e.g. Fig. 2.4.) □

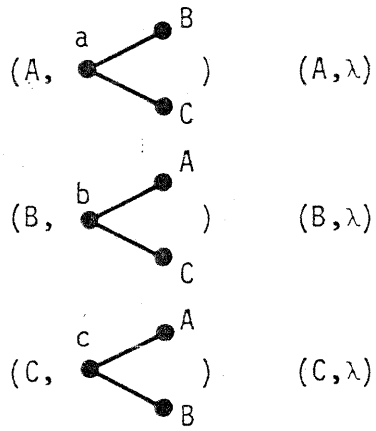


Fig. 2.2. Production set P_2 of BNLC grammar G_2.

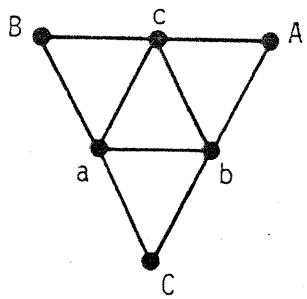


Fig. 2.3. The start graph Z_2 of BNLC grammar G_2.

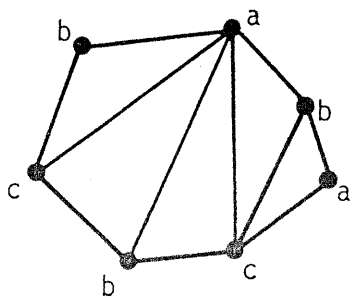


Fig. 2.4. A (labeled) maximal outerplanar graph in $L(G_2)$.

A small modification of grammar G_2 yields a grammar G_3 which generates all 2-trees (see Rose, 1974). 2-trees are unlabeled graphs which are defined recursively as follows. (1) The triangle is the only 2-tree on three nodes. (2) If $\{x,y\}$ is an edge in a 2-tree, then the unlabeled graph obtained by adding a new node z connected to x and y is a 2-tree. (3) Only those unlabeled graphs obtained by (1) and (2) are 2-trees.

Example 3 (2-trees). Let $G_3 = (\Sigma_3,\Delta_3,P_3,\underline{conn}_3,Z_3)$, where $\Delta_3 = \Delta_2$, $\Sigma_3 = \Sigma_2$, $\underline{conn}_3 = \underline{conn}_2$, $Z_3 = Z_2$, and the productions of P_3 are depicted in Fig. 2.5. It is not too difficult to see how the productions in P_3 simulate the recursive build-up of a 2-tree. Hence, $\underline{und}(L(G_3))$ consists of all 2-trees. □

The reader who is interested in k-trees (see Rose, 1974) might easily find a BNLC grammar which generates all k-trees for arbitrary but fixed positive k, so e.g. all trees which coincide with 1-trees.

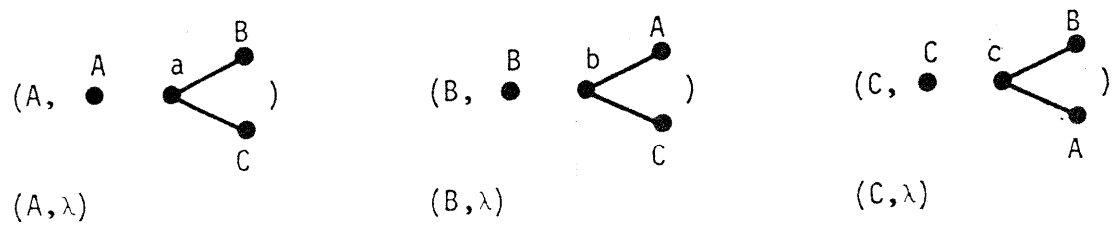


Fig. 2.5. Production set P_3 of BNLC grammar G_3.

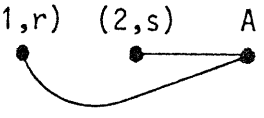
The last example of this section is somewhat more involved than the previous ones. However, it will lead us to a lower bound for the complexity of underlying unlabeled languages of BNLC languages. A similar example has already been considered in Turán (1983) for NLC grammars.

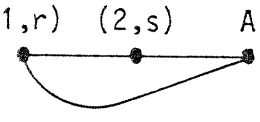
The cyclic bandwidth of a graph X is the minimum integer k for which there is a bijection f from V_X to $\{1,2,\ldots,\#X\}$, such that for all $\{x,y\} \in E_X$, $\min(|f(x) - f(y)|, \#X - |f(x) - f(y)|) \leq k$. That is, there is a cyclic order of V_X such that adjacent nodes are separated by at most k-1 nodes in this order. A bijection f satisfying the above property is called a cyclic bandwidth k layout of X.

Example 4 (graphs of cyclic bandwidth ≤ 2). Let $\Delta_4 = \{(i,r) \mid i \in \{1,2,1',2'\}$ and $r \subseteq \{1,2,1',2'\}\}$ and let $\Sigma_4 = \Delta_4 \cup \{A_0, A\}$. Consider the BNLC grammar $G_4 = (\Sigma_4, \Delta_4, P_4, \underline{conn}_4, Z_4)$, where $\underline{conn}_4((i,r)) = \{(j,s) \mid j \in r, s \subseteq \{1,2,1',2'\}\}$ for $(i,r) \in \Delta_4$, $\underline{conn}_4(A_0) = \emptyset$ and $\underline{conn}_4(A) = \{(1',\emptyset),(2',\emptyset)\}$, Z_4 is a single node labeled by A_0.

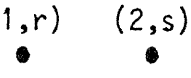
P_4 consists of the following four types of productions.

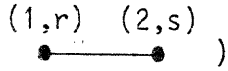
(a) Initializing productions: $(A_0, X) \in P_4$, for all graphs X which can be obtained from the graph (with bold edges) in Fig. 2.6 by adding zero or more of the dotted edges.

(b) Constructing productions: $(A,$ 

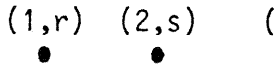
$) \in P_4$ and $(A,$ 

$) \in P_4$ for all $r \subseteq \{1,2\}$ and all $s \subseteq \{2\}$.

(c) Even terminating productions: $(A,$ 

$) \in P_4$ and $(A,$ 

$) \in P_4$ for all $r \subseteq \{1,2,1'\}$ and all $s \subseteq \{2,1',2'\}$.

(d) Odd terminating productions: $(A,$ 

$) \in P_4$ for all $r \subseteq \{1,2\}$, $s \subseteq \{2,1'\}$, and $t \subseteq \{1',2'\}$, and all productions are in P_4 which can be obtained from these productions by adding edges in the right-hand sides.

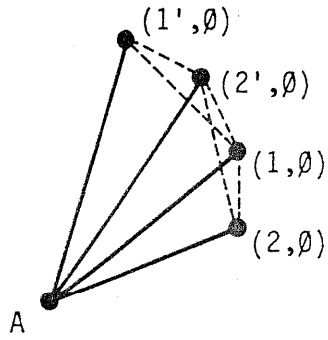


Fig. 2.6. Basic graph for initializing productions in P_4

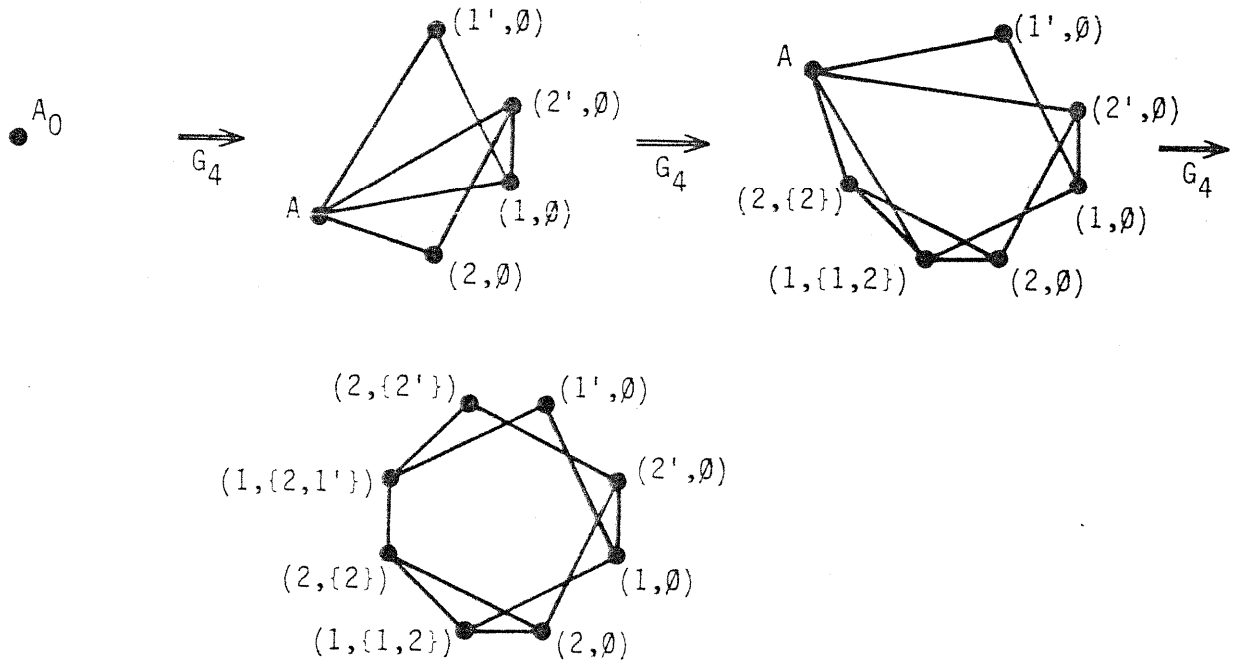


Fig. 2.7. A derivation of a graph of cyclic bandwidth ≤ 2 in G_4.

It is an easy exercise to follow the derivations of graphs in $L(G_4)$ (see e.g. Fig. 2.7) and to observe that these graphs always have cyclic bandwidth ≤ 2 (and at least 6 nodes). On the other hand, let X be an unlabeled graph of cyclic bandwidth ≤ 2 with at least 6 nodes and let f be a cyclic bandwidth 2 layout of X. Then one can use the cyclic order to build up a derivation of a graph X' in G_4 such that $X = \underline{und}(X')$. For the labeling one proceeds as follows: the nodes $f^{-1}(1)$, $f^{-1}(2)$, $f^{-1}(3)$, and $f^{-1}(4)$, respectively are labeled by $(1',\emptyset)$, $(2',\emptyset)$, $(1,\emptyset)$, and $(2,\emptyset)$, respectively. Then the first components of the labels are chosen 1 and 2 alternatingly. The second components are chosen, such that the connections backwards (and for the last two or three nodes also forwards) are established correctly. For example, if $\#X \geq 8$, then $f^{-1}(5)$ is labeled by $(1,r)$, where $r \subseteq \{1,2\}$, $1 \in r$ if and only if $f^{-1}(5)$ is adjacent to $f^{-1}(3)$, and $2 \in r$ if and only if $f^{-1}(5)$ is adjacent to $f^{-1}(4)$. Thus $\underline{und}(L(G_4))$ consists of all unlabeled graphs of cyclic bandwidth ≤ 2 with at least 6 nodes. □

We state now a number of basic observations on BNLC grammars. Some of the technical lemmas we state in the rest of this section are so basic in the considerations on BNLC grammars that they are often used implicitly in the proofs (and so not referred to explicitly).

Consider a BNLC grammar $G = (\Sigma,\Delta,P,\underline{conn},Z)$. If we apply a production of P to a Γ-boundary graph then, obviously, the resulting graph is a Γ-boundary graph. Since Z is a Γ-boundary graph, we get:

LEMMA 2.1. Let $G = (\Sigma,\Delta,P,\underline{conn},Z)$ be a BNLC grammar. Then every graph in S(G) is a Γ-boundary graph. □

As a consequence of this lemma, for every BNLC grammar $G = (\Sigma,\Delta,P,\underline{conn},Z)$, there exists a BNLC grammar $G' = (\Sigma,\Delta,P,\underline{conn}',Z)$ with $L(G) = L(G')$, such that $\underline{conn}'(d) \subseteq \Delta$, for all $d \in \Sigma$. G' can be obtained simply by setting $\underline{conn}'(d) = \underline{conn}(d) \cap \Delta$ for all $d \in \Sigma$, because a mother node is never adjacent to

a nonterminal node and hence connections between the daughter graph and nonterminal nodes cannot be established.

On the other hand, let $G = (\Sigma,\Delta,P,\underline{conn},Z)$ be an NNLC grammar such that $\underline{conn}(d) \subseteq \Delta$, for all $d \in \Sigma$. Consider now a graph X in a derivation of a graph in L(G) where two nonterminal nodes x and y are adjacent. Then, in some subsequent step, either x or y must be replaced. In either case all connections between the daughter graph replacing x, say, and the node y are broken, because $\varphi_X(y) \notin \underline{conn}(\varphi_X(z))$ for all z of the daughter graph. That is, edges between nonterminal nodes are never used to establish a connection in a derivation in G. Hence, if we erase all edges between nonterminal nodes in Z and in the right-hand side of each production in P, then this does not affect the generated language. (Of course, it affects the exhaustive language!) Thus, we can construct a BNLC grammar G' with L(G) = L(G'). Consequently, by setting a restriction on the connection function of an NNLC grammar one gets an elegant characterization of BNLC languages.

THEOREM 2.2. A graph language L is a BNLC language if and only if there is an NNLC grammar $G = (\Sigma,\Delta,P,\underline{conn},Z)$ with $\underline{conn}(d) \subseteq \Delta$ for all $d \in \Sigma$, such that $L = L(G)$. □

Another consequence of Lemma 2.1 is that the order of applying productions in a derivation of a BNLC grammar does not affect the resulting graph of this derivation. More precisely, suppose we apply a production to a node x and a production to a node y in a graph X in the exhaustive language of a BNLC language. Then we obtain the same graph, independently of the fact whether we replace x first and then y or the other way round. This is stated formally in the following lemma.

LEMMA 2.3. Let G be a BNLC grammar. Let $X_0 \in S(G)$, let $x,y \in V_{X_0}$ and let Y_1,Y_2,X_1,X_2 be graphs such that

$$X_0 \underset{G}{\Rightarrow} (x,Y_1) \quad X_1 \underset{G}{\Rightarrow} (y,Y_2)\ X_2$$

holds. If X_1' and X_2' are the graphs, such that

$$X_0 \underset{G}{\Rightarrow} (y,Y_2) \quad X_1' \underset{G}{\Rightarrow} (x,Y_1) \quad X_2'$$

holds, then $X_2' = X_2$. □

The property described in Lemma 2.3 is commonly referred to as "finite Church Rosser property".

We conclude this section by providing technical tools which are crucial in forthcoming proofs.

Concrete derivations

Let $G = (\Sigma,\Delta,P,\underline{conn},Z_{ax})$ be an NLC grammar. If a graph X concretely derives a graph Z in G replacing a node x by a graph Y, then, somewhat informally, we refer to the construct $X \Rightarrow_{(x,Y)} Z$ as a concrete derivation step (in G) from X to Z. (To be more formal one would have to use the rather tedious notation of 5-tuples (X,x,Y,Z,G).)

A sequence of "successive" concrete derivation steps in G,

$$D: \quad X_0 \Rightarrow_{(x_0,Y_1)} X_1 \Rightarrow_{(x_1,Y_2)} X_2 \cdots \Rightarrow_{(x_{n-1},Y_n)} X_n,$$

$n \geq 0$, where the sets V_{X_0} and V_{Y_i}, $1 \leq i \leq n$, are pairwise disjoint, is referred to as a concrete derivation (in G) from X_0 to X_n.

The node set of D is $V_D = \bigcup_{i=0}^{n} V_{X_i}$. The edge set of D is $E_D = \bigcup_{i=0}^{n} E_{X_i}$. The labeling function φ_D of D is defined by $\varphi_D(x) = \varphi_{X_0}(x)$ if $x \in V_{X_0}$ and $\varphi_D(x) = \varphi_{Y_i}(x)$ if $x \in V_{Y_i}$ for some i, $1 \leq i \leq n$. Note that $V_D = V_{X_0} \cup \bigcup_{i=1}^{n} V_{Y_i}$, hence φ_D is defined on the whole set V_D. Moreover, if $x \in V_{X_i}$ for some i, $0 \leq i \leq n$, then $\varphi_{X_i}(x) = \varphi_D(x)$. Thus every concrete derivation D defines naturally a graph with set of nodes V_D, set of edges E_D and labeling function φ_D; this justifies our abuse of notation in using V_D, E_D, and φ_D when referring to various elements of a concrete derivation D. Note that this "graph" D is a Γ-boundary graph whenever X_0

is a Γ-boundary graph and G is a BNLC grammar.

The following basic property (use) of concrete derivations is given without a proof.

LEMMA 2.4. Let $G = (\Sigma,\Delta,P,\underline{conn},Z_{ax})$ be a BNLC grammar. Then a graph X is in L(G) if and only if there is a concrete derivation D in G from a graph $X_0 \in [Z_{ax}]$ to $X \in G_\Delta$. □

Let D be a concrete derivation as described above and let $\mathcal{O}_D$ be a distinguished element not in V_D which is called the origin of D. The predecessor mapping $\underline{pred}_D$ of D is a function from V_D into $V_D \cup \{\mathcal{O}_D\}$, such that for $x \in V_D$

$$\underline{pred}_D(x) = \begin{cases} \mathcal{O}_D & \text{if } x \in V_{X_0}, \quad \text{and} \\ x_i & \text{if } x \in V_{Y_{i+1}} \quad \text{for an } i,\ 0 \leq i \leq n-1. \end{cases}$$

Hence $\underline{pred}_D$ maps every node x in V_D to the node from which x is directly derived (or to $\mathcal{O}_D$ if x was already present in X_0).

The history $\underline{hist}_D(x)$ of a node x in V_D is the sequence $\underline{hist}_D(x) = (y_0,y_1,\ldots,y_m)$, $m \geq 1$, $y_i \in V_D$ for i, $1 \leq i \leq m$, such that $y_0 = \mathcal{O}_D$, $y_m = x$, and $y_i = \underline{pred}_D(y_{i+1})$ for all i, $0 \leq i \leq m-1$.

Finally, we denote the set of nodes in V_{X_n} which are derived from a node $x \in V_D$ by $\underline{targ}_D(x)$, that is $\underline{targ}_D(x) = \{y \in V_{X_n} \mid x \in \underline{hist}_D(y)\}$.

(For a sequence $s = (y_0,y_1,\ldots,y_n)$ we write $x \in s$ if there is an i, $0 \leq i \leq m$, such that $x = y_i$).

The usefulness of the above notions stems from the following result.

LEMMA 2.5. Let $G = (\Sigma,\Delta,P,\underline{conn},Z_{ax})$ be a BNLC grammar and for a Γ-boundary graph X_0 let

$$D: X_0 \Rrightarrow_{(x_0,Y_1)} X_1 \Rrightarrow_{(x_1,Y_2)} X_2 \cdots \Rrightarrow_{(x_{n-1},Y_n)} X_n$$

be a concrete derivation in G (for convenience we set $Y_0 = X_0$).

(i) If $\{y,z\} \in E_D$ then $\underline{pred}_D(y) \in \underline{hist}_D(z)$ or $\underline{pred}_D(z) \in \underline{hist}_D(y)$.

(ii) Let $y,z \in V_D$ be such that for $\underline{hist}_D(y) = (y_0,y_1,\ldots,y_m)$, $m \geq 1$, there is a k, $0 \leq k \leq m-1$, with $\underline{pred}_D(z) = y_k$. Then $\{y,z\} \in E_D$ if and only if $\{y_{k+1},z\} \in E_{Y_i}$ for some i, $0 \leq i \leq n$, and $\varphi_D(z) \in \underline{conn}(\varphi_D(y_i))$ for all i, $k+2 \leq i \leq m$.

Proof. (i) Let $\{y,z\} \in E_D$. Consider $\underline{hist}_D(y) = (y_0,y_1,\ldots,y_m)$ and $\underline{hist}_D(z) = (z_0,z_1,\ldots,z_\ell)$. Let i be the minimal index such that $y_i \neq z_i$. Clearly, $i \geq 1$, because $z_0 = y_0 = \mathcal{O}_D$. Obviously, if $\{y_m,z_\ell\} \in E_D$, then $\{y_i,z_i\} \in E_D$. Consequently, either $\varphi_D(y_i) \in \Delta$ or $\varphi_D(z_i) \in \Delta$ (see Lemma 2.1). Suppose $\varphi_D(z_i) \in \Delta$, which implies that $z_i = z_\ell = z$. Hence $\underline{pred}_D(z) = z_{i-1} = y_{i-1} \in \underline{hist}_D(y)$. Similarly, if $\varphi_D(y_i) \in \Delta$, then $\underline{pred}_D(y) \in \underline{hist}_D(z)$.

(ii) Clearly, for i, $k+2 \leq i \leq m$, $\{y_i,z\} \in E_D$, if and only if $\{y_{i-1},z\} \in E_D$ and $\varphi_D(z) \in \underline{conn}(\varphi_D(y_i))$. Since $\underline{pred}_D(z) = \underline{pred}_D(y_{k+1})$, there is an i, $0 \leq i \leq n$, such that y_{k+1} and z are in V_{Y_i}. That is, $\{y_{k+1},z\} \in E_D$ if and only if $\{y_{k+1},z\} \in E_{Y_i}$. □

A BNLC grammar $G = (\Sigma,\Delta,P,\underline{conn},Z_{ax})$ is <u>normalized</u> if (1) for all $(A,Y) \in P$, $\#Y \geq 1$, (2) $\#Z_{ax} = 1$, and (3), for all $d \in \Sigma$, $\underline{conn}(d) \subseteq \Delta$. Using standard techniques, the following normal form can be shown (see also the discussion before Theorem 2.2).

THEOREM 2.6. For every BNLC language L there is a normalized BNLC grammar G with $L(G) = L - \{\lambda\}$. □

In what follows we consider two graph languages to be <u>equal</u> if they coincide up to the empty graph λ; two BNLC grammars are called <u>equivalent</u> if the languages they generate are equal.

3. CONTEXT CONSISTENT NORMAL FORM

One essential feature of a BNLC grammar is that in a derivation the neighborhood of a nonterminal node never changes, until the node itself is rewritten. This fact, together with the following general property of NLC grammars will be exploited in many proofs of this paper.

LEMMA 3.1. Let $X \Rightarrow_{(x,Y)} Z$ be a concrete derivation step in an NLC grammar $G = (\Sigma,\Delta,P,\underline{conn},Z_{ax})$. For a node $y \in V_Y$, its context in Z can be expressed by by $\underline{cont}_Z(y) = \underline{cont}_Y(y) \cup (\underline{cont}_X(x) \cap \underline{conn}(\varphi_Y(y)))$. □

Let $G = (\Sigma,\Delta,P,\underline{conn},Z_{ax})$ be an NNLC grammar. G is context consistent, if there is a function η from Γ into 2^Σ with the following property. For every graph $X \in S(G)$ and for every nonterminal node $x \in V_X$, $\underline{cont}_X(x) = \eta(\varphi_X(x))$ holds. The function η satisfying the above is called the context describing function of G.

THEOREM 3.2. For every BNLC language L there is a normalized context consistent BNLC grammar G such that $L = L(G)$.

Proof. Let $G = (\Sigma,\Delta,P,\underline{conn},Z_{ax})$ be a BNLC grammar. We will construct an equivalent context consistent BNLC grammar $G' = (\Sigma',\Delta,P',\underline{conn}',Z'_{ax})$. For the construction we exploit the following facts: (i) The context of a node of a daughter graph can be calculated (as indicated by Lemma 3.1) from the context of the mother node and (ii) the context of a nonterminal node does not change until the node itself is rewritten.

Let $\Gamma' = \Gamma \times 2^\Delta$ and let $\Sigma' = \Gamma' \cup \Delta$. Each element of Γ' will be of the form A_r, where $A \in \Gamma$ and $r \subseteq \Delta$. Intuitively a node labeled by A_r (in a graph from $S(G')$) will correspond to a node labeled by A with context r (in a graph from $S(G)$). The relabeling ρ from Σ' to Σ is defined by $\rho(A_r) = A$ for $A_r \in \Gamma'$, and $\rho(a) = a$ for $a \in \Delta$. The new connection function $\underline{conn}'$ is defined by $\underline{conn}'(d) = \underline{conn}(\rho(d))$ for all $d \in \Sigma'$. Z'_{ax} is obtained from Z_{ax} by replacing each nonterminal label A of a node x in Z_{ax} by A_r, where $r = \underline{cont}_{Z_{ax}}(x)$. Finally, the

set of production P' is defined as follows. Let $(A,Y) \in P$ and let $r \subseteq \Delta$. Then P' contains a production (A_r,Y') such that (1) $\rho(Y') = Y$, and (2) if, for a node $y \in V_Y$, $\varphi_Y(y) = B \in \Gamma$, then $\varphi_{Y'}(y) = B_s$, where $s = \underline{cont}_Y(y) \cup (r \cap \underline{conn}(B))$. No other productions but those defined above are in P'.

Let η be the function from Γ into 2^Δ defined by $\eta(A_r) = r$ for all $A_r \in \Gamma'$ It is easy to see that G' is context consistent and that η is the context describing function of G'.

Moreover, if we take a concrete derivation D' in G' and we replace very label $A_r \in \Gamma'$ by A, then we get a concrete derivation D in G, which implies that $L(G') \subseteq L(G)$.

On the other hand, consider a concrete derivation

$$D: X_0 \Rrightarrow_{(x_0,Y_1)} X_1 \Rrightarrow_{(x_1,Y_2)} X_2 \cdots \Rrightarrow_{(x_{n-1},Y_n)} X_n$$

in G with $X_0 \in [Z_{ax}]$ and $X_n \in L(G)$. If we replace the label A of each nonterminal node x in a graph X_i, for some i, $0 \leq i \leq n$, by the label $A_r \in \Gamma'$ with $r = \underline{cont}_{X_i}(x)$, then this induces in the obvious way the labeling of the nodes in the graphs Y_i, $1 \leq i \leq n$. (Note that if $x \in V_{X_i}$, $x \in V_{X_j}$, $0 \leq i \leq j \leq n$, and $\varphi_D(x) \in \Gamma$, then $\underline{cont}_{X_i}(x) = \underline{cont}_{X_j}(x)$). It is not too difficult to see that this new labeling yields a concrete derivation D' in G' from a graph $X'_0 \in [Z'_{ax}]$ to X_n. Hence $L(G) \subseteq (G')$ and we have $L(G) = L(G')$.

Note that if G is normalized, then so is G'. Since we may assume that G is normalized (Theorem 2.6), the theorem follows. □

The following extension of an NLC grammar has been considered in Janssens & Rozenberg (1980b).

A context-sensitive NLC grammar G is a system $G = (\Sigma,\Delta,P,\underline{conn},Z_{ax})$ where Σ, Δ, $\underline{conn}$ and Z_{ax} are defined as in the case of an NLC grammar and P is a finite set of triples (d,Y,C), $d \in \Sigma$, $Y \in G_\Sigma$ and $C \subseteq \Sigma$. A production (d,Y,C) is applicable to a node x in a graph X if $\varphi_X(x) = d$ and $C \subseteq \underline{cont}_X(x)$. If a production (d,Y,C) is applicable, then its application is identical to the application of the

production (d,Y) in an NLC grammar. The language L(G) and the exhaustive language S(G) of G are defined in the same way as in the case of an NLC grammar. $G = (\Sigma,\Delta,P,\underline{conn},Z_{ax})$ is a context-sensitive BNLC grammar if it is a context sensitive NLC grammar and $G' = (\Sigma,\Delta,P',\underline{conn},Z_{ax})$, with $P' = \{(d,Y) \mid (d,Y,C) \in P$, for some $C \subseteq \Sigma\}$, is a BNLC grammar.

It has been shown in Janssens & Rozenberg (1980b) that there are context-sensitive NLC grammars G such that L(G) is not an NLC language. Using the ideas from the proof of the preceding theorem, we show that the context-sensitive extension does not affect the generative power of BNLC grammars.

THEOREM 3.3. A graph language L is generated by a context-sensitive BNLC grammar G if and only if it is generated by a BNLC grammar G'.

Proof. (sketch). Let $G' = (\Sigma',\Delta',P',\underline{conn}',Z')$ be a BNLC grammar. Clearly, if we replace every production $(d,Y) \in P'$, by a context-sensitive production $(d,Y,\emptyset)$, then we obtain an equivalent context-sensitive BNLC grammar $G = (\Sigma',\Delta',P,\underline{conn}',Z')$.

On the other hand, let $G = (\Sigma,\Delta,P,\underline{conn},Z)$ be a context-sensitive BNLC grammar. Now let us modify G analogously as it was done in the proof of Theorem 3.2 to obtain G'. That is, $\Sigma' = (\Gamma \times 2^{\Delta}) \cup \Delta$, conn and Z are appropriately changed to conn' and Z', respectively, and for every production (A,Y,C) in P and for every subset $r \in \Delta$, we get in P' a corresponding production (A_r,Y',C). (As a matter of fact, the reason that we have not required at the very beginning of the proof of Theorem 3.2 that G is normalized is that in this way the proof "applies" also to a "general" context-sensitive BNLC grammar). The so obtained context-sensitive BNLC grammar $G' = (\Sigma',\Delta,P',\underline{conn}',Z')$ is equivalent to G, i.e., L(G) = L(G') and whenever a production (A_r,Y',C) is applied to a node x in a graph $X \in S(G')$ then $\underline{cont}_X(x) = r$. Now it is not too difficult to see, that, if we define $P'' = \{(A_r,Y') \mid (A_r,Y',C) \in P'$ and $C \subseteq r\}$, then the BNLC grammar $G'' = (\Sigma',\Delta,P'',\underline{conn}',Z')$ generates the language L(G") = L(G') = L(G). □

4. NEIGHBORHOOD PRESERVING NORMAL FORM

Consider a concrete derivation step $X \Rightarrow_{(x,Y)} Z$ in an NLC grammar G. While the graph Z is uniquely determined by X and (x,Y), there might be several graphs X' such that $X' \Rightarrow_{(x,Y)} Z$ holds. That is, even if we specify the daughter graph Y in Z and the production we use, there are still many possible steps "backwards". This stems essentially from the fact that nodes in X which are adjacent to x might establish no connection to any node in Y. This feature of NLC grammars is an obvious serious drawback for efficient bottom-up parsing.

On the other hand, if we can assume that x is adjacent only to these nodes in X which are adjacent to some node of Y in Z, i.e., if $\underline{neigh}_X(x) = \underline{neigh}_Z(V_Y)$, then X is uniquely determined by Z and (x,Y). In this section we prove a normal form for BNLC grammars which - in deriving graphs from its language - use "uniquely invertible" concrete derivation steps only.

Note that e.g. in Rosenfeld & Milgram (1972) this property was obtained by forcing an applicability condition on the node to be replaced in order to ease efficient parsing techniques.

Let G be an NLC grammar. A concrete derivation step $X \Rightarrow_{(x,Y)} Z$ in G is <u>neighborhood preserving</u> if $\underline{neigh}_X(x) = \underline{neigh}_Z(V_Y)$. An NLC grammar G is <u>neighborhood preserving</u> if every concrete derivation step from X to Z in G with $X \in S(G)$ is neighborhood preserving.

It is not too difficult to see that it is decidable whether a BNLC grammar is neighborhood preserving. Moreover, in the case of context consistent BNLC grammars one gets the following grammatical characterization. (In what follows $\underline{lab}(G)$ denotes the set of all labels occurring in graphs from S(G)).

LEMMA 4.1. A context consistent BNLC grammar $G = (\Sigma,\Delta,P,\underline{conn},Z_{ax})$ with context describing function η is neighborhood preserving if and only if, for every production $(A,Y) \in P$, where $A \in \underline{lab}(G)$, we have

$\eta(A) \subseteq \bigcup_{y \in V_Y} \underline{conn}(\varphi_Y(y))$. □

THEOREM 4.2. For every BNLC language L there is a normalized neighborhood preserving and context consistent BNLC grammar such that $L = L(G)$.

Proof. Let L be a BNLC language and let $G = (\Sigma,\Delta,P,\underline{conn},Z_{ax})$ be a normalized context consistent BNLC grammar with context describing function η such that $L = L(G)$. (The existence of G follows from Theorem 3.2.)

We will define a BNLC grammar $G' = (\Sigma',\Delta,P',\underline{conn}',Z'_{ax})$, such that for every concrete derivation

$$D: X_0 \Rightarrow_{(x_0,Y_1)} X_1 \Rightarrow_{(x_1,Y_2)} X_2 \cdots \Rightarrow_{(x_{n-1},Y_n)} X_n$$

in G with $X_0 \in [Z_{ax}]$, there is a corresponding derivation

$$D': X'_0 \Rightarrow_{(x_0,Y'_1)} X'_1 \Rightarrow_{(x_1,Y'_2)} X'_2 \cdots \Rightarrow_{(x_{n-1},Y'_n)} X'_n$$

in G' with $X'_0 \in [Z'_{ax}]$, such that edges in D which are "broken" in a later step are not established in D'. That is, each graph X'_i, $0 \leq i \leq n$, is a copy of the graph X_i, $0 \leq i \leq n$, except that all edges from X_i that are broken later on in D are erased.

The construction of G' is based on the following technique. Each label A can "guess" which edges (incident with a node labeled by A) will and which will not be used later on. Thus, to every label A in a graph a subset r of Δ is added. Intuitively speaking, this subset r encodes the following information: a node labeled by A_r has not established edges to a-labeled nodes for all $a \in r$, although "it should have" (following the original grammar G).

G' is formally defined as follows. Let $\Gamma' = \{ A_r \mid A \in \Gamma,\ r \subseteq \eta(A) \}$ and let $\Sigma' = \Gamma' \cup \Delta$. We use the relabeling ρ from Σ' to Σ, defined by $\rho(a) = a$ for $a \in \Delta$, and $\rho(A_r) = A$ for $A_r \in \Gamma'$. For $A_r \in \Gamma'$, let $\underline{conn}'(A_r) = \underline{conn}(A) - r$, and for $a \in \Delta$, let $\underline{conn}'(a) = \underline{conn}(a)$. If A is the label of the unique node of Z_{ax}, then Z'_{ax} is a node labeled by $A_\emptyset$.

In order to define P' we need the following additional concepts. Consider a

Γ-boundary graph Y. A function φ' from V_Y into Σ' is called a transfer labeling of φ_Y if $\rho(\varphi'(y)) = \varphi_Y(y)$ for all $y \in V_Y$. A transfer labeling φ' of φ_Y defines a φ'-transfer graph Y' of Y by $V_{Y'} = V_Y$, $\varphi_{Y'} = \varphi'$, and

$$E_{Y'} = E_Y - \{\{y,z\} | \varphi'(y) = A_r \in \Gamma' \text{ and } \varphi'(z) \in r\}.$$

(Intuitively speaking, we obtain Y' from Y by replacing each nonterminal A of a node y by A_r for some $r \subseteq n(A)$, and by deleting all edges between y and all neighbors that are labeled by some $a \in r$.)

Let (A,Y) be a production in P. A production (A_r,Y') iscalled a transfer production of (A,Y) if the following conditions hold.

(i) $A_r \in \Gamma'$,

(ii) Y' is the $\varphi_{Y'}$-transfer graph of Y,

(iii) if, for some $y \in V_Y$, $\varphi_{Y'}(y) = B_s \in \Gamma'$, then $s \supseteq r \cap \underline{conn}(B)$ holds,

(iv) $n(A) - r \subseteq \bigcup_{y \in V_{Y'}} \underline{conn}'(\varphi_{Y'}(y))$, and

(v) if, for $y \in V_{Y'}$, $\varphi_{Y'}(y) = a \in \Delta$, then $\underline{conn}'(a) \cap r = \emptyset$.

Now P' is defined by

$$P' = \{(A_r,Y') | (A_r,Y') \text{ is a transfer production of some } (A,Y) \in P\}.$$

Roughly speaking, the basic intuition underlying the imposition of conditions (i) through (v) above is as follows. As mentioned already, if a node x is labeled by A_r, then this means that x is "guessing" that, for $a \in r$, connections to a-labeled nodes will be broken (in G) in a later step, and so they do not have to be established (in G') at all. This makes sense only if $r \subseteq n(A)$ which is then the only assumption for A_r to be in Γ'. Clearly, (to make the guess "correct") all connections of a node $y \in V_Y$, labeled by B_s to nodes labeled by some $a \in s$ must be broken which yields condition (ii). (Recall that, for $a \in \Delta$, a node derived from y establishes connections either to all a-labeld neighbors of y or to none of them.)

If, for a label $a \in \Delta$, $a \in r$ and $a \in \underline{conn}(\varphi_Y(y))$ for some node $y \in V_{Y'}$, then y is forced to disconnect from all a-labeled nodes because it cannot connect to the former a-labeled neighbors of the replaced node (labeled by A_r); here "former" means "in the original grammar". That is, for $\varphi_{Y'}(y) = B_s$, a must be in s which explains condition (iii). Condition (iv) is needed to make the grammar G' neighborhood preserving (see Lemma 4.1).

Finally, condition (v) enables "to check" whether the "guesses" of the predecessors of a terminal node in Y' were correct, and it prevents the grammar from deriving graphs with some edges "missing" (in G').

To prove that indeed $L(G) = L(G')$ and that G' is neighborhood preserving, we go through the following four claims.

Claim 1. G' is context consistent with the context describing function η' defined by $\eta'(A_r) = \eta(A) - r$ for $A_r \in \Gamma'$.

Proof of Claim 1. Let $X \in L(G')$ be such that $\underline{cont}_X(x) = \eta'(\varphi_X(x))$ for all nonterminal nodes x of X. Consider a concrete derivation step $X \Rightarrow_{(x,Y)} Z$ in G'. Then $\underline{cont}_Z(z) = \underline{cont}_X(z)$ for all nonterminal nodes z in V_{X-x}; hence η' describes the context of these nodes.

Let $\varphi_X(x) = A_r \in \Gamma'$, and let y be a nonterminal node of Y, that is, $\varphi_Y(y) = B_s$ for some $B_s \in \Gamma'$. Then (by Lemma 3.1), $\underline{cont}_Z(y) = \underline{cont}_Y(y) \cup (\eta'(A_r) \cap \underline{conn}'(B_s))$. Let (A_r, Y) be a transfer production of $(A, \bar{Y}) \in [P]$. Then we know that $\eta(B) = \underline{cont}_{\bar{Y}}(y) \cup (\eta(A) \cap \underline{conn}(B))$. Since $\eta'(A_r) = \eta(A) - r$, $\underline{conn}'(B_s) = \underline{conn}(B) - s$, and $\underline{cont}_Y(y) = \underline{cont}_{\bar{Y}}(y) - s$, (and, moreover, $s \supseteq r \cap \underline{conn}(B)$), it follows that $\underline{cont}_Z(y) = \eta(B) - s = \eta'(B_s)$. This proves the claim, because η' describes the context of the unique node of Z'_{ax}.

Claim 2. G' is neighborhood preserving.

Proof of Claim 2. The claim follows from Claim 1, from the definition (see point (iv)) of a transfer production, and from Lemma 4.1.

Claim 3. $L(G') \subseteq L(G)$.

Proof of Claim 3. We show that every graph $X' \in S(G')$ is the $\varphi_{X'}$-transfer graph of a graph $X \in S(G)$. Since a graph X' with $\underline{lab}(X') \subseteq \Delta$ is the $\varphi_{X'}$-transfer graph of a graph X if and only if $X = X'$, this will prove the Claim.

Obviously, Z'_{ax} is the $\varphi_{Z'_{ax}}$-transfer graph of Z_{ax}.

Consider a graph $X' \in S(G')$ which is the $\varphi_{X'}$-transfer graph of a graph $X \in S(G)$. Let $X' \Rightarrow_{(x,Y')} Z'$ be a concrete derivation step in G', where $(\varphi_{X'}(x),Y') \in [P']$ is a transfer production of $(A,Y) \in [P]$. Hence, $\varphi_{X'}(x) = A_r$ for some $r \subseteq \Gamma$, and, consequently, $\varphi_X(x) = A$. Thus $X \Rightarrow_{(x,Y)} Z$ is a concrete derivation step in G, for the appropriate graph Z. Our goal is to show that Z' is the $\varphi_{Z'}$-transfer graph of Z. Obviously, $V_{Z'} = V_{Z'}$ and $\varphi_{Z'}$ is a transfer labeling of $\varphi_{Z'}$ because $\varphi_{Y'}$ is a transfer labeling of φ_Y and $\varphi_{X'-x}$ is a transfer labeling of φ_{X-x}. It is left to show that

$$E_{Z'} = E_Z - \{\{y,z\} \mid \varphi_{Z'}(y) = B_s \in \Gamma' \text{ and } \varphi_{Z'}(z) \in s\}.$$

Clearly, only edges $\{y,z\}$ with $y \in V_{Y'}$ and $z \in V_{X'-x}$ are crucial. Due to condition (v) from the definition of a transfer production, all such edges $\{y,z\} \in E_Z$ with $\varphi_Z(y) \in \Delta$ and $\varphi_Z(z) \in \Delta$ are also present in $E_{Z'}$.
On the one hand if, for some $y \in V_{y'}$, $\varphi_{Z'}(y) = B_s \in \Gamma'$ and, for some $z \in V_{X'-x}$, $\varphi_{Z'}(z) = a \in s$, then $\{y,z\} \notin E_{Z'}$, because $a \notin \underline{conn}'(B_s) = \underline{conn}(B) - s$. On the other hand, let $\{y,z\} \in E_Z$, $y \in V_{Y'}$, $\varphi_{Z'}(y) = B_s \in \Gamma'$, and $z \in V_{X'-x}$, $\varphi_{Z'}(z) = a \in \Delta - s$. Then $a \in \underline{conn}(B)$ and $a \in \underline{conn}'(B_s)$. Moreover, due to condition (iii) on the transfer production (A_r,Y'), we have $a \notin r$. Consequently, since $\{x,z\} \in E_X$, we have $\{x,z\} \in E_X$, and so $\{y,z\} \in E_{Z'}$. This shows that Z' is the $\varphi_{Z'}$-transfer graph of Z, which proves the claim.

Claim 4. $L(G) \subseteq L(G')$.

Proof of Claim 4. Consider a concrete derivation,

$$D: X_0 \Rightarrow_{(x_0,Y_1)} X_1 \Rightarrow_{(x_1,Y_2)} X_2 \cdots \Rightarrow_{(x_{n-1},Y_n)} X_n$$

in G from $X_0 \in [Z_{ax}]$ to $X_n \in L(G)$.

An edge $\{x,y\} \in E_D$ is called <u>productive</u>, if there exist nodes $x' \in \underline{targ}_D(x)$ and $y' \in \underline{targ}_D(y)$ such that $\{x',y'\} \in E_{X_n}$. An edge in E_D is <u>non-productive</u> if it is not productive.

Clearly, if $\{x,y\} \in E_D$ is non-productive, then exactly one of the nodes x and y is a nonterminal node. Moreover, we make the following basic observation: let $x,y,z \in V_D$ with $\varphi_D(x) \in \Gamma$ and $\varphi_D(y) = \varphi_D(z) \in \Delta$. Provided $\{x,y\} \in E_D$ and $\{x,z\} \in E_D$, $\{x,y\}$ is productive if and only if $\{x,z\}$ is productive. This stems essentially from the fact that a node derived from x cannot distinguish between y and z, because they have the same label.

We will "modify" D in a way such that we get a concrete derivation D' in G' from a graph $X_0' \in [Z_{ax}']$ to X_n. To this end, we define a function φ' from V_D into Σ' as follows. If $\varphi_D(x) \in \Delta$, then $\varphi'(x) = \varphi_D(x)$. If $\varphi_D(x) = A \in \Gamma$, then $\varphi'(x) = A_r$, where

$$r = \{a \in \Delta \mid \text{there is a non-productive edge } \{x,y\} \in E_D \text{ with } \varphi_D(y) = a\}$$

Obviously, $A_r \in \Gamma'$. This function φ' induces transfer labelings for all graphs X_i, $0 \le i \le n$, and Y_j, $1 \le j \le n$. By this we obtain a φ'-transfer graph X_i' for each X_i, $0 \le i \le n$, and a φ'-transfer graph Y_j for each Y_j, $1 \le j \le n$. (Formally, we would have to take the appropriate restrictions of φ' in each case!)

We claim that the sequence

$$D': X_0' \Rightarrow_{(x_0,Y_1)} X_1' \Rightarrow_{(x_1,Y_2)} X_2' \cdots \Rightarrow_{(x_{n-1},Y_n')} X_n'$$

is a concrete derivation in G'. Since, obviously, $X_n' = X_n$ and $X_0' \in [Z_{ax}']$, this will prove that $X_n \in L(G')$. We concentrate on the fact that $(\varphi'(x_i),Y_{i+1}')$ is a transfer production of $(\varphi_D(x_i),Y_{i+1})$, for all i, $0 \le i \le n-1$. The rest follows easily.

Let us fix an i, $0 \le i \le n-1$, and let $\varphi'(x_i) = A_r$. Since $A_r \in \Gamma'$ and since Y_{i+1}' is the φ'-transfer graph of Y_{i+1}, <u>conditions</u> (i) <u>and</u> (ii) on a transfer pro-

duction are satisfied.

Suppose condition (iii) is violated for some y of Y'_{i+1} with $\varphi_{Y'_{i+1}}(y) = B_s \in \Gamma'$. That is, there is a terminal label $a \in r \cap \underline{conn}(B)$, such that $a \notin s$. If $a \in r$, then there is a node z in V_D with $\{x_i,z\} \in E_D$, $\varphi_D(z) = a$ and $\{x_i,z\}$ is non-productive. Thus, if $a \in \underline{conn}(B)$, then $\{y,z\} \in E_D$, and $\{y,z\}$ is non-productive (otherwise $\{x_i,z\}$ would be productive, because $\underline{targ}_D(y) \subseteq \underline{targ}_D(x_i)$). This contradicts the fact that $a \notin s$ and, consequently, condition (iii) holds.

To show that condition (iv) from the definition of a transfer production is fulfilled, let $a \in n(A) - r$. That is, there is a productive edge $\{x_i,z\} \in E_{X_i}$ with $\varphi_{X_i}(z) = a$, because $a \in n(A) = \underline{cont}_{X_i}(x_i)$ and $a \notin r$. Since $\{x_i,z\}$ is productive, there is a node y of Y_{i+1} such that $\{y,z\} \in E_D$ and $\{y,z\}$ is productive. This implies that $a = \varphi_D(z) \in \underline{conn}(\varphi_Y(y))$. If $\varphi_y(y) \in \Delta$, then $\underline{conn}'(\varphi_{Y'_{i+1}}(y)) = \underline{conn}(\varphi_{Y_{i+1}}(y))$ and condition (iv) is satisfied. If $\varphi_{Y_{i+1}}(y) = B \in \Gamma$, then $\varphi_{Y'_{i+1}}(y) = B_s \in \Gamma'$ for an appropriate $s \subseteq \Delta$. Since $\{y,z\}$ is productive, every edge $\{y,z'\}$ with $\varphi_D(z') = a$ is productive (recall the above observation) and $a \notin s$. Consequently, $a \in \underline{conn}'(\varphi_{Y_{i+1}}(y))$, and condition (iv) is satisfied in this case, too.

Finally we check that condition (v) holds. This can be done easily, since, if $\{x_i,z\} \in E_{X_i}$ is non-productive and $\varphi_D(z) = b \in \Delta$, then no terminal node y of Y_{i+1} connects to b, i.e., $b \notin \underline{conn}(\varphi_{Y_{i+1}}(y))$ for all nodes y of Y_{i+1} with $\varphi_{Y_{i+1}}(y) \in \Delta$. Hence, if $b \in r$, then $b \notin \underline{conn}'(a)$ for all $a \in \underline{lab}(Y'_{i+1}) \cap \Delta$.

Hence conditions (i) through (v) from the definition of a transfer production hold and we can conclude that $(\varphi'(x_i),Y_{i+1})$ is a transfer production of $(\varphi_D(x_i),Y_{i+1})$. It is now easily seen that D' is a concrete derivation in G' from a graph $X'_0 \in [Z'_{ax}]$ to X_n which proves Claim 4.

The theorem follows from Claims 1 through 4. □

5. RESTRICTIONS ON THE RIGHT-HAND SIDES

In this section we investigate the existence of normal forms for BNLC grammars which impose specific restrictions on the form of the right-hand sides of productions. In particular, we are interested in the following restrictions: (i) if the right-hand side of a production consists of one node only, then it must be a terminal node, and (ii) the number of nodes of the right-hand side graph of each production does not exceed k, where k is a fixed constant (for all BNLC grammars).

It turns out that the normal form for BNLC grammars satisfying (i) above exists, and that for no k one can obtain a normal form satisfying (ii) above.

It is instructive to compare these results with results from Welzl (1984) and Ehrenfeucht et al. (1984) - in the first of these papers it is demonstrated that the normal form for all NLC grammars satisfying (i) above does not exist, while in Ehrenfeucht et al. (1984) it is demonstrated that there is also no normal form of type (ii) for all NLC grammars. (Actually, it turns out that this second result can be easily carried over to BNLC grammars.)

Let $G = (\Sigma,\Delta,P,\underline{conn},Z)$ be an NNLC grammar. A production $(A,Y) \in P$ is called a chain-rule, if $V_Y = \{y\}$ and y is a nonterminal node.

We would like to point out, that the standard construction of eliminating chain-rules from context-free string grammars (see, e.g., Salomaa, 1973, Theorem 6.3) cannot be applied directly to BNLC grammars. Consider, e.g., the productions $(A,\bullet^B)$ and $(B,\bullet^C)$, A and B nonterminals and c a terminal. Moreover, let $\underline{conn}(A) = \underline{conn}(c) = \{b\}$ and let $\underline{conn}(B) = \emptyset$. Then, in the standard construction, we would introduce $(A,\bullet^C)$ to "simulate" the successive application of the above two productions. However, if an A-labeled node x is adjacent to a b-labeled node z, then (1) after applying $(A,\bullet^C)$ to x, the c-labeled node is adjacent ot z, while (2) after applying $(A,\bullet^B)$ and then $(B,\bullet^C)$, the c-labeled node is not adjacent to z. Nevertheless, using the neighborhood preserving normal form result

in Theorem 4.2, one can prove the following theorem.

THEOREM 5.1. For every BNLC language L there is a normalized neighborhood preserving and context consistent BNLC grammar G without chain-rules such that $L = L(G)$.

Proof. Let L be a BNLC language. Then there is a normalized neighborhood preserving and context consistent BNLC grammar $G = (\Sigma, \Delta, P, \underline{conn}, Z)$ such that $L = L(G)$. Let η be the context describing function of G.

Consider a chain-rule $(A,Y) \in P$, where $A \in \underline{lab}(G)$, $V_Y = \{y\}$ and $\varphi_Y(y) = B \in \Gamma$. Since G is neighborhood preserving, $\underline{conn}(B) \supseteq \eta(A)$ and, consequently, $\eta(B) = \eta(A)$. Of course, we can assume that $\underline{conn}(B) \subseteq \eta(B)$, so that we get $\underline{conn}(B) = \eta(B) = \eta(A)$. Hence we can assume that the application of a chain-rule to a graph in S(G) amounts to a simple relabeling (without changing the edges). Thus we can follow the ideas of the standard construction for eliminating chain-rules from context-free string grammars.

For $A,B \in \Gamma$, we say A directly chain-derives B, if there is a chain-rule $(A, \bullet^B) \in P$. A chain-derives B, if there is a sequence $A_0, A_1, \ldots, A_n$, $n \geq 0$, $A_0 = A$, $A_n = B$, such that A_{i-1} directly chain-derives A_i, for all i, $1 \leq i \leq n$.

Let $P' = \{(A,Y) \mid (B,Y) \in P$, A chain-derives B, for some $B \in \Gamma$, and (A,Y) is no chain-rule$\}$ and let $G' = (\Sigma, \Delta, P', \underline{conn}, Z)$. Then, using the above reasoning, it is not too difficult to prove that $L(G') = L(G)$; it is also obvious that G' is normalized, neighborhood preserving and context consistent, and that G' has no chain-rules. □

As an immediate consequence of the above normal form, we get the following enumeration result for BNLC languages which can be proved along the lines of the proof from Welzl (1984).

For a graph language L and a positive integer n, we denote by $\underline{numb}_L(n)$ the number of nonisomorphic graphs in L with up to n nodes, that is,

$\underline{numb}_L(n) = \#\{[X] \mid X \in L$ and $\#X \leq n\}$.

THEOREM 5.2. For every BNLC language L there is a positive integer c such that $\underline{numb}_L(n) \leq 2^{c.n}$, for all positive integers n. □

It is known, that for a label b, the set of all graphs over {b} is an NLC language. Clearly for this language the assertion of Theorem 5.2 does not hold. Hence we get the following result.

COROLLARY 5.3. The family of (u-)BNLC languages is a proper subset of the family of (u-)NLC languages. □

Let a be a fixed label. For $k \geq 3$, we denote by C_k the cycle of length k with all nodes labeled by a.

PROPOSITION 5.4. (Ehrenfeucht et al., 1984). For each $k \geq 1$, the graph language $L = \{C_{k+3}\}$ cannot be generated by an NLC grammar G with $\underline{maxr}(G) \leq k$. □

Since every finite graph language is a BNLC language, for every k, $\{C_{k+3}\}$ is a BNLC language. Hence we get the following corollary.

COROLLARY 5.5. For every positive integer k, there is a BNLC language L which cannot be generated by a BNLC grammar G with $\underline{maxr}(G) \leq k$. □

The corresponding problem for u-BNLC languages is open, i.e., we do not know whether there is a positive integer k_0, such that for every u-BNLC language L there is a BNLC grammar G with $\underline{maxr}(G) \leq k_0$ and $L = \underline{und}(L(G))$.

6. COMPLEXITY OF BOUNDARY NLC GRAMMARS

We consider the complexity of BNLC languages, relabeled BNLC languages and u-BNLC languages. For unexplained complexity notions we refer to Garey & Johnson (1979).

A result in Janssens & Rozenberg (1980b, Theorem 9) shows that the membership problem for NLC languages is PSPACE-complete (see also Brandenburg, 1983, Theorem 2). For BNLC languages Theorem 5.1 yields immediately the following "better" upper bound.

THEOREM 6.1. The membership problem for BNLC languages and for u-BNLC languages is in NP. □

Note, however, that there are NLC grammars G without chain-rules, such that the membership problem for L(G) is NP-complete (as shown by Turán, 1983 and by Brandenburg, 1983).

We analyze a bottom-up (dynamic programming) parsing technique for BNLC languages which can be shown to work in polynomial time for connected graphs of (fixed) bounded degree. This result will be generalized to relabeled BNLC languages i.e., to graph languages $\rho(L)$, where L is a BNLC language and ρ is a relabeling.

Our parsing technique relies essentially on the following two properties of BNLC grammars: (1) the finite Church Rosser property (as formulated in Lemma 2.3) and (2) the existence of the neighborhood preserving normal form (Theorem 4.2). A number of ideas in subsequent proofs were inspired by proof techniques from Slisenko (1982).

Let Σ and Δ be sets of labels such that $\Sigma \subseteq \Delta$ and let X be a graph over Δ. Consider a subset $R = \{(d_1,V_1),(d_2,V_2),\ldots,(d_m,V_m)\}$ of $\Sigma \times 2^{V_X}$, such that $V_i \cap V_j = \emptyset$ for all i,j, $1 \leq i < j \leq m$. Then we set $V(R) = \bigcup_{i=1}^{m} V_i$ and we define the graph Z(R) over Σ by $V_{Z(R)} = R \cup \{(\varphi_X(x),\{x\}) \mid x \in \underline{neigh}_X(V(R))\}$,

$$E_{Z(R)} = \{\{(d,V),(d',V')\} \mid (d,V),(d',V') \in V_{Z(R)},\ V \neq V',\ \text{and there are } y \in V,\ y' \in V' \text{ with } \{y,y'\} \in E_X\},$$

and, for all $(d,V) \in V_{Z(R)}$, $\varphi_{Z(R)}((d,V)) = d$. The graph Y(R) is the subgraph of Z(R) induced by R. (Intuitively speaking, Z(R) can be obtained from X by (i) deleting all nodes not in $V(R) \cup \underline{neigh}_X(V(R))$ and (ii) "contracting" the nodes in each of the sets V_i, $1 \leq i \leq m$, to one node labeled by d_i.)

Let $G = (\Sigma,\Delta,P,\underline{conn},Z_{ax})$ be a neighborhood preserving BNLC grammar and let X be a graph over Δ. The bottom-up parsing set of X in G, bup(G,X) for short, is a subset of $\Sigma \times 2^{V_X}$, and it is defined recursively as follows:

(1) For each $x \in V_X$, $(\varphi_X(x),\{x\}) \in \underline{bup}(G,X)$.

(2) Let $R = \{(d_1,V_1),(d_2,V_2),\ldots,(d_m,V_m)\}$ be a subset of bup(G,X) such that $V_i \cap V_j = \emptyset$, for all i,j, $1 \leq i < j \leq m$. If there is a label $d \in \Sigma$, such that

$$Z(\{(d,V(R))\}) \Rightarrow_{((d,V(R)),Y(R))} Z(R)$$

is a neighborhood preserving concrete derivation step, then $(d,V(R)) \in \underline{bup}(G,X)$, and we write $(d,V(R)) \rightarrow R$.

(3) No other elements but those obtained by steps (1) and (2) are in bup(G,X).

LEMMA 6.2. Let G be a normalized neighborhood preserving BNLC grammar and let A_0 be the label of the unique node of the axiom of G. A graph X is in L(G) if and only if (A_0,V_X) is in bup(G,X).

Proof. Let $G = (\Sigma,\Delta,P,\underline{conn},Z_{ax})$. First we show that if $X \in L(G)$ then $(A_0,V_X) \in \underline{bup}(G,X)$. To this aim consider a concrete derivation,

$$D: X_0 \Rightarrow_{(x_0,Y_1)} X_1 \Rightarrow_{(x_1,Y_2)} \cdots \Rightarrow_{(x_{n-1},Y_n)} X_n$$

in G from $X_0 \in [Z_{ax}]$ to $X = X_n$. We claim that for all i, $0 \leq i \leq n-1$, $(\varphi_D(x_i), \underline{targ}_D(x_i)) \in \underline{bup}(G,X)$. Since $\varphi_D(x_0) = A_0$ and $\underline{targ}_D(x_0) = V_X$, this will prove the above assertion.

By definition, $(\varphi_D(x),\underline{targ}_D(x)) \in \underline{bup}(G,X)$ for all $x \in V_X$. Let k, $0 \leq k \leq n-1$, be a positive integer such that $(\varphi_D(x_i),\underline{targ}_D(x_i)) \in \underline{bup}(G,X')$, for all i, $k+1 \leq i \leq n-1$, and consider the concrete derivation step $X_k \Rightarrow_{(x_k,Y_{k+1})} X_{k+1}$.

Let X_k' be the subgraph of X_k which is induced by $\{x_k\} \cup \underline{neigh}_{X_k}(x_k)$ and let X_{k+1}' be the subgraph of X_{k+1} which is induced by $V_{Y_{k+1}} \cup \underline{neigh}_{X_{k+1}}(V_{Y_{k+1}})$. Since G is neighborhood preserving, we have $\underline{neigh}_{X_k}(x_k) = \underline{neigh}_{X_{k+1}}(V_{Y_{k+1}})$ and, consequently,

$$X_k' \Rightarrow_{(x_k,Y_{k+1})} X_{k+1}'$$

is a neighborhood preserving concrete derivation step in G.

Let $R = \{(\varphi_D(x),\underline{targ}_D(x)) \mid x \in V_{Y_{k+1}}\}$. Then, by our assumption, $R \subseteq \underline{bup}(G,X)$. Moreover, it is easily seen that $Z(R)$ is isomorphic to X_{k+1}', $Y(R)$ is isomorphic to Y_{k+1}, and $Z(V(R))$ is isomorphic to X_k'. (Take the isomorphism f defined by $f(x) = (\varphi_D(x), \underline{targ}_D(x))$ for all $x \in V_{X_k} \cup V_{X_{k+1}}$). Hence it is easily seen that $(\varphi_D(x_k),\underline{targ}_D(x_k)) \in \underline{bup}(G,X)$, which proves that $(\varphi_D(x_i),\underline{targ}_D(x_i)) \in \underline{bup}(G,X)$, for all i, $0 \leq i \leq n-1$.

Secondly, we show that if $(A_0,V_X) \in \underline{bup}(G,X)$, then $X \in L(G)$. Note that if $(d,V) \in \underline{bup}(G,X)$, then either $d \in \Delta$ and $\#V = 1$ or there is a subset R of $\underline{bup}(G,X)$ such that $(d,V) \rightarrow R$, which implies that $(d,Y(R)) \in [P]$. This relation can be used for a build-up of a concrete derivation,

$$D: X_0 \Rightarrow_{(x_0,Y_1)} X_1 \Rightarrow_{(x_1,Y_2)} X_2 \cdots \Rightarrow_{(x_{n-1},Y_n)} X_n$$

in G from $X_0 = Z(\{(A_0,V_X)\})$ to $X_n = Z(\{(\varphi_X(x),\{x\}) \mid x \in V_X\})$ as follows. For each i, $i = 1,2,\ldots,n$, let $Y_i = Y(R)$, where $R \subseteq \underline{bup}(G,X)$ is such that $x_{i-1} \rightarrow R$. (Note that the nodes x_i involved are now elements from $\Sigma \times 2^{V_X}$.) The proof that D is really a concrete derivation in G is not difficult and so it is left to the reader. Obviously, $X_n \in [X]$ which proves that $X \in L(G)$. □

THEOREM 6.3. Let L be a fixed BNLC language and let k be a fixed natural number. Then membership in L of connected graphs of maximal degree not exeeding k can be decided in polynomial time.

Proof. Let $G = (\Sigma,\Delta,P,\underline{conn},Z_{ax})$ be a neighborhood preserving normalized BNLC grammar such that $L = L(G)$ (see Theorem 4.2). Since there is only a polynomial number of subsets R of $\underline{bup}(G,X)$ with $1 \leq \#R \leq \underline{maxr}(G)$. (G is considered fixed), it is easily seen that $\underline{bup}(G,X)$ can be computed in time polynomial in $\#\underline{bup}(G,X)$. Hence, it remains to show that the size of $\underline{bup}(G,X)$ is polynomial in #X, where X is a connected graph of maximal degree $\leq k$.
This will be done in the following three claims.

Claim 1. If $(d,V) \in \underline{bup}(G,X)$, then $\#\underline{neigh}_X(V) \leq k \cdot \#\Delta$.

Proof of Claim 1. Let $R_0 = \{(d,V)\}$ and let $R = \{(\varphi_X(x),\{x\})|x \in V\}$. Then the graphs $Z(R_0)$ and $Z(R)$ are well defined and it can be easily seen that there is a concrete derivation D in G from $Z(R_0)$ to $Z(R)$, consisting of neighborhood preserving steps only (recall the construction in the proof of Lemma 6.2). Hence, for every terminal label a, there is a node in Z(R) which is adjacent to all a-labeled neighbors of (d,V) in $Z(R_0)$. Thus, (d,V) has at most k a-labeled neighbors in $Z(R_0)$, which implies that there are at most $k \cdot \#\Delta$ neighbors of (d,V) in $Z(R_0)$. Obviously, $\#\underline{neigh}_{Z(R_0)}((d,V)) = \#\underline{neigh}_X(V)$ and so the claim holds.

Claim 2. If $(d,V) \in \underline{bup}(G,X)$, then $\#\{\{x,y\} \in E_X|x \in V_X - V, y \in V\} \leq k^2 \cdot \#\Delta$.

Proof of Claim 2. There are at most $k \cdot \#\Delta$ nodes in $\underline{neigh}_X(V)$ and each of these nodes has at most k incident edges which yields the inequality in the claim.

For a subset V of V_X let $\underline{nv}(V) = \underline{neigh}(V)$ and let $\underline{ne}(V) = \{\{x,y\} \in E_X|x \in V_X - V, y \in V\}$. Then Claims 1 and 2 imply that there is only a polynomial number of different sets $\underline{nv}(V)$ and $\underline{ne}(V)$, if we consider all sets V with $(d,V) \in \underline{bup}(G,X)$ for some $d \in \Sigma$. Hence the following claim will settle our problem.

Claim 3. Let $(d,V),(d,V') \in \underline{bup}(G,X)$. If $\underline{nv}(V) = \underline{nv}(V')$ and $\underline{ne}(V) = \underline{ne}(V')$, then $V = V'$.

Proof of Claim 3. Since X is a connected graph, there is a path from every node to every node in X. Now let V_0 be the set of all nodes x in V_X, such that each path from x to a node $y \in \underline{nv}(V)$ must pass an edge in $\underline{ne}(V)$. It is not difficult to see that $V_0 = V$ and so V is completely determined by $\underline{nv}(V)$ and $\underline{ne}(V)$. This proves the claim.

In summary, we have shown that there are only a polynomial number of sets V which appear in elements of $\underline{bup}(G,X)$ - this shows that the cardinality of $\underline{bup}(G,X)$ is polynomial in #X. Since this implies that we can compute $\underline{bup}(G,X)$ in time polynomial in #X and since, by Lemma 6.2, $X \in L(G)$ if and only if $(A_0,V_X) \in \underline{bup}(G,X)$ (where A_0 is the unique label of Z_{ax}), the theorem holds. □

We extend now our result to polynomial time membership of connected graphs of bounded (fixed) degree in relabeled BNLC languages, i.e.,in languages of the form $\rho(L)$, where L is a BNLC language and ρ is a relabeling. This will be done using a technique similar as above. Now, however, we have to find a derivation of a graph in a BNLC grammar together with a relabeling which yields the desired result.

Let $G = (\Sigma,\Delta,P,conn,Z_{ax})$ be a neighborhood preserving BNLC grammar, let ρ be a relabeling from Δ to a set of labels Δ' and let X be a graph over Δ'.

The bottom-up ρ-relabeled parsing set of X in G, $\underline{bupr}(\rho,G,X)$ for short, consists of triples (d,V,μ) where $d \in \Sigma$, $V \in V_X$ and μ is a function from $\underline{neigh}_X(V)$ into Δ', such that for $x \in \underline{neigh}_X(V)$, $\rho(\mu(x)) = \varphi_X(x)$. That is, for every node x in $\underline{neigh}_X(V)$, μ guesses a label $a = \mu(x)$ which is consistent with φ_X and ρ in the sense that $\mu(a) = \varphi_X(x)$. Whenever below we consider triples (d,V,ξ) we assume that either ξ is a function μ as above or ξ is a "dummy" (place holder) which we will denote by $\langle\rangle$.

Let $R = \{(d_1,V_1,\mu_1),(d_2,V_2,\mu_2),\ldots,(d_m,V_m,\mu_m)\}$ be a set of triples as

described above. Then R is consistent, if the following conditions hold.

(i) $V_i \cap V_j = \emptyset$ for all i,j, $1 \leq i < j \leq m$

(ii) For all i, $1 \leq i \leq m$, $\mu_i \neq \langle\rangle$.

(iii) If $x \in \underline{neigh}_X(V_i)$ and $x \in \underline{neigh}_X(V_j)$ for some i,j, $1 \leq i < j \leq m$, then $\mu_i(x) = \mu_j(x)$.

(iv) If $x \in \underline{neigh}_X(V_i)$, $V_j = \{x\}$ and $d_j \in \Delta$ for some i,j, $1 \leq i,j \leq m$, then $\mu_i(x) = d_j$.

If R is consistent, then we set $\bar{V}(R) = \bigcup_{i=1}^{m} V_i$ and we define the graph $\bar{Z}(R)$ over Σ by

$$V_{\bar{Z}(R)} = R \cup \{(d,\{x\},\langle\rangle) \mid x \in \underline{neigh}_X(V_i) - \bar{V}(R),\ d = \mu_i(x),\ \text{for some } i,\ 1 \leq i \leq m\},$$

$$E_{\bar{Z}(R)} = \{\{(d,V,\mu),(d',V',\mu')\} \mid V \neq V' \text{ and there are } y \in V,\ y' \in V' \text{ with } \{y,y'\} \in E_X\},$$

and for $(d,V,\mu) \in V_{\bar{Z}(R)}$, $\varphi_{\bar{Z}(R)}((d,V,\mu)) = d$. The graph $\bar{Y}(R)$ is the subgraph of $\bar{Z}(R)$ induced by R.

The set $\underline{bupr}(\rho,G,X)$ is now defined as follows.

(1) For every $x \in V_X$, $(a,\{x\},\mu) \in \underline{bupr}(\rho,G,X)$ for all $a \in \Delta$ with $\rho(a) = \varphi_X(x)$, and for all functions μ from $\underline{neigh}_X(x)$ into Δ, such that, for all $y \in \underline{neigh}_X(x)$, $\rho(\mu(y)) = \varphi_X(y)$.

(2) Let R be a consistent subset of $\underline{bupr}(\rho,G,X)$. If there is a label d such that

$$\bar{Z}(\{(d,\bar{V}(R))\}) \Rightarrow_{((d,\bar{V}(R)),\bar{Y}(R))} \bar{Z}(R)$$

is a neighborhood preserving concrete derivation step in G, then $(d,\bar{V}(R)) \in \underline{bupr}(\rho,G,X)$.

(3) No other elements but those defined by steps (1) and (2) are in $\underline{bupr}(\rho,G,X)$.

Although bupr(ρ,G,X) is somewhat more involved than bup(G,X), it is clear that it "represents" a bottom-up parsing technique for $\rho(L(G))$ in the same way as bup(G,X) represents a bottom-up parsing technique for L(G). Hence, we can state the following lemma (the proof of the lemma is left to the reader, because it can be made analogous to the proof of Lemma 6.2).

LEMMA 6.4. Let G be a normalized neighborhood preserving BNLC grammar, let A_0 be the label of the unique node of the axiom of G and let ρ be a relabeling from the set of terminal labels Δ of G into a set of labels Δ'.
(i) For a graph $X \in G_{\Delta'}$, $X \in \rho(L(G))$ if and only if $(A_0,V_X,\mu_\emptyset) \in$ bupr(ρ,G,X), where $\mu_\emptyset$ is the (in this case) unique function with empty domain.
(ii) If G and ρ are fixed, then bupr(ρ,G,X) can be computed in time polynomial in #bupr(ρ,G,X). □

THEOREM 6.5. Let L be a fixed BNLC language, let k be a fixed positive integer and let ρ be a relabeling. Then the membership of a connected graph of maximal degree not exceeding k in $\rho(L)$ can be decided in polynomial time.

Proof. Let $G = (\Sigma,\Delta,P,\text{conn},Z)$ be a normalized neighborhood preserving BNLC grammar such that $L = L(G)$, let ρ be a relabeling from Δ into a set of labels Δ' and let $X \in G_{\Delta'}$ be a connected graph of maximal degree $\leq k$.

From the proof of Theorem 6.3, we know that there are only a polynomial number of different sets $V \subseteq V_X$ which appear in triples $(d,V,\rho) \in$ bupr(ρ,G,X). Moreover, if $(d,V,\mu) \in$ bupr(ρ,G,X), then $\#\text{neigh}_X(V) \leq k \cdot \#\Delta$. Hence, once V is determined there are at most $(\#\Delta')^{k\cdot\#\Delta}$ different possible functions μ such that $(d,V,\mu) \in$ bupr(ρ,G,X). This shows that the cardinality of bupr(ρ,G,X) is polynomial in #X and so,by Lemma 6.4,the theorem holds. □

Every u-BNLC language can be regarded as a relabeled BNLC language which entails the following theorem.

THEOREM 6.6. Let L be a fixed u-BNLC language and let k be a fixed positive integer. Then membership (in L) of connected unlabeled graphs of maximal degree not exceeding k can be decided in polynomial time. □

We would like to make here two explanatory remarks concerning the above result.

On the one hand, the reader might argue that the assumptions (connected and bounded degree) from the statement of the above theorem are too restrictive to provide interesting applications. However, the theorem is strong enough to yield, e.g. polynomial time recognition of graphs with bandwidth $\leq k$, binary tree bandwidth $\leq k$ or cutwidth $\leq k$ and of connected graphs with cyclic bandwidth $\leq k$ for fixed k (grammars for these graph languages are not difficult to obtain). The last mentioned example, i.e., polynomial time recognition of connected graphs with cyclic bandwidth $\leq k$, is the best possible one can expect, unless P = NP, because this problem becomes NP-complete for disconnected graphs and k = 2 (see Leung et al., 1984). Actually, this NP-completeness result will be used in two theorems below to demonstrate, that our result for u-BNLC languages is indeed on the boundary between "polynomial time" and "NP-complete".

On the other hand, the result should be regarded as a tool for showing - in a first step - that a graph language is polynomial time recognizable. Since the constants (in the exponent) we obtain are "very big" (we did not even elaborate them in detail), our algorithms cannot replace tailormade efficient algorithms, as, e.g. the $O(n^k)$ recognition algorithm of Gurari & Sudborough (1982) for graphs of bandwidth $\leq k$ (n is the size of the graph under consideration).

THEOREM 6.7. There is a u-BNLC language L of bounded degree, such that the membership problem for L is NP-complete.

Proof. Recall Example 4 from Section 2. There a BNLC grammar G_4 is given, such that $\underline{und}(L(G_4))$ consists of all unlabeled graphs of cyclic bandwidth ≤ 2 with at least 6 nodes. Clearly, the degree of a node in a graph with cyclic bandwidth ≤ 2 does not exceed 4. Hence, $L(G_4)$ is of bounded degree.

It is known that the problem of deciding whether a graph has cyclic bandwidth ≤ 2 is NP-complete (see Leung et al., 1984). Hence the language $\underline{und}(L(G_4))$ is NP-complete. □

A graph language L is of bounded average degree if there is an integer c such that $\#E_X \leq c \cdot \#X$ holds for all graphs X in L. (That is, the average of the degrees of the nodes in each graph in L is bounded by a common constant.)

THEOREM 6.8. There is a u-BNLC language L of bounded average degree which contains only connected graphs such that the membership problem in L is NP-complete.

Proof. Consider the BNLC grammar $G = (\Sigma,\Delta,P,\underline{conn},Z)$ which is obtained from $G_4 = (\Sigma_4,\Delta_4,P_4,\underline{conn}_4,Z_4)$ in Example 4 of Section 2 as follows.

$\Delta = \Delta_4 \cup \{¢\}$, where $c \notin \Delta_4$, $\Sigma = \Sigma_4 \cup \{¢\}$, and $P = P_4$. For $d \in \Sigma_4$, $\underline{conn}(d) = \underline{conn}_4(d) \cup \{¢\}$, and $\underline{conn}(¢) = \emptyset$. Finally, Z consists of two adjacent nodes, one node labeled by A and the other node labeled by ¢. Clearly, L(G) consists of all graphs which can be obtained from the graphs in $L(G_4)$ by adding a node labeled by ¢ which is adjacent to all other nodes. That is $\underline{und}(L(G))$ contains all unlabeled graphs (with at least 7 nodes) consisting of an unlabeled graph of bandwidth ≤ 2 together with a node adjacent to all other nodes. Hence, $\underline{und}(L(G))$ is of bounded average degree and it contains connected graphs only, and it is easily seen that $\underline{und}(L(G_4))$ is polynomial time reducible to $\underline{und}(L(G))$. □

7. DISCUSSION

In this paper we have presented some basic properties of the class of BNLC grammars. This class seems to be definitionally "attractive" (natural).

(A1) On the one hand it has been introduced by requiring that all the graphs involved in the definition of an BNLC grammar (i.e., the axiom and the right-hand sides of the productions) are Γ-boundary graphs, where Γ is the set of non-terminal labels involved.

(A2) On the other hand the class of BNLC languages may be defined by imposing a natural restriction on the connection functions used in NLC grammars.

It appears that the family of BNLC grammars enjoys quite attractive properties. For example:

(B1) A number of interesting families of graphs can be generated using BNLC grammars. (We have given here grammars for some of these families mentioned in the introduction.)

(B2) Interesting and clearly technically useful normal form results have been established for the class of BNLC grammars.

(B3) We have demonstrated that for a fixed positive integer k and for a fixed (u-)BNLC language L, the question whether or not an arbitrary connected (unlabeled) graph of degree not exceeding k belongs to L is solvable in polynomial time.

We see this paper as the initial step in a systematic investigation of BNLC grammars and languages. It turns out that the class of BNLC languages enjoys more interesting properties. For example, one can prove the following: "Whenever we can find for a set of graphs a BNLC grammar which generates it, then for graphs of (fixed) bounded degree in this language, k-colorability and the subgraph homeomorphism problem (with respect to a fixed graph) can be solved in polynomial time". A consequence of this result is e.g. that k-colorability is solvable in polynomial time for graphs of bounded cyclic bandwidth. This demonstrates that one

can get nontrivial complexity results simply by providing a grammar for a set of graphs. The above assertion and other results concerning BNLC grammars will be presented in a following paper.

The present paper points out to a number of topics and technical problems for further investigations.

(C1) We have seen that BNLC grammars satisfy the "Church Rosser property", i.e., the order of applying the productions in a derivation does not influence the final result. The relationship to other subclasses of NLC grammars satisfying the Church Rosser property should be investigated (for some of these subclasses see Ehrig et al., 1982, and Janssens, 1983).

(C2) We know that for no fixed k, BNLC grammars G with $\underline{maxr}(G) \leq k$ can generate all BNLC languages. Does this statement still hold, if one considers u-BNLC languages?

Acknowledgements: The first author acknowledges the support by NSF Grant MCS 83-05245. Research of the second author has been partially supported by the Austrian "Fonds zur Förderung der wissenschaftlichen Forschung".

REFERENCES

Brandenburg, F.J. (1983), On the complexity of the membership problem for graph grammars, in "Proceedings of the WG 83" (Nagl, M. & Perl, J, eds.), Universitätsverlag Trauner, Linz, pp. 40-49

Ehrenfeucht, A., Main, M.G. & Rozenberg, G. (1984), Restrictions on NLC grammars, Theoret.Comput.Sci. 31, pp. 211-223.

Ehrig, H. (1979), Introduction to the algebraic theory of graph grammars, Lecture Notes in Computer Science 73, pp. 1-69.

Ehrig, H.,Janssens, D.,Kreowski, H.-J. & Rozenberg, G. (1982), Concurrency of node labelled graph transformations, University of Antwerp, Report 82-38.

Garey, M.R. & Johnson, D.S. (1979), "Computers and Intractability - A Guide to the Theory of NP-completeness", Freeman, San Francisco.

Gurari, E.H. & Sudborough, I.H. (1982), Improved dynamic programming algorithms for bandwidth minimization and the min-cut linear arrangement problem, to appear in J. Algorithms.

Harary, F. (1969), "Graph Theory", Addison Wesley, Reading, Massachusetts.

Janssens, D. (1983), "Node label Controlled Graph Grammars", Ph.D. Thesis, University of Antwerp.

Janssens, D. & Rozenberg, G. (1980a), On the structure of node label controlled graph languages, Inform. Sci. 20, pp. 191-216.

Janssens, D. & Rozenberg, G. (1980b), Restrictions, extensions and variations of NLC grammars, Inform. Sci. 20, pp. 217-244.

Janssens, D. & Rozenberg, G. (1981), Decision problems for node label controlled graph grammars, J. Comp. System Sci. 22, pp. 144-174.

Janssens, D., Rozenberg, G. & Welzl, E. (1984), The bounded degree problem for NLC grammars is decidable, Institute of Applied Mathematics and Computer Science, University of Leiden, Report Nr. 84-05.

Leung, J.Y.-T.,Witthof,J. & Vornberger, O. (1984), On some variations of the bandwidth minimization problem, SIAM J. Comput. 13, pp. 650-667.

Nagl, M. (1979), A tutorial and bibliographical survey of graph grammars, Lecture Notes in Computer Science 73, pp. 70-126.

Proskurowski, A. (1978), Minimum dominating cycles in 2-trees, Internat. J. Comput. Inform. Sci. 8, pp. 405-417.

Rose, D.J. (1974), On simple characterizations of k-trees, Discrete Math. 7, pp. 317-320.

Rosenfeld, A. & Milgram, D. (1972), Web automata and web grammars, Machine Intelligence 7, pp. 307-324.

Salomaa, A. (1973), "Formal Languages", Academic Press, London.

Slisenko, A.O. (1982), Context-free grammars as a tool for describing polynomial-time subclasses of hard problems. Inform. Process. Lett. 14, pp. 52-56.

Turán, Gy. (1983), On the complexity of graph grammars, Acta Cybernet. 6, 271-281.

Welzl, E. (1984), Encoding graphs by derivations and implications for the theory of graph grammars, Lecture Notes in Computer Science 172, pp. 503-513.